CHAPTER 2

Exploring the UNIX File System

case Dominion Consulting is creating a centralized telephone database that will contain each employee's name and phone number information. Currently each department maintains this information and stores it in multiple files located in separate directories. Your job is to consolidate this information in a single master database that everyone in the company can access.

LESSON A

objectives

After studying this lesson, you should be able to:

- **Discuss and explain the UNIX file system**
- **Define a UNIX file system partition**
- **Use the mount command to mount a file system**
- **Discuss relative and absolute path addressing**
- **Diagram the UNIX file system hierarchy**
- **Navigate the file system**

Understanding Files and Directories

In this chapter, you explore the UNIX file system, including the basic concepts of files and directories and their organization in a hierarchical tree structure. After you learn to navigate the file system, you practice what you've learned by creating directories and files and copying files from one directory to another. You also have the opportunity to set file permissions, which is important in a multi-user system like UNIX.

Understanding the UNIX File System

In UNIX, a **file** is the basic component for data storage. UNIX considers everything it interacts with as a file, even attached devices such as the monitor, keyboard, and printer. A **file system** is the UNIX system's way of organizing files on mass storage devices such as hard and floppy disks. A physical file system is a section of the hard disk that has been formatted to hold files. UNIX consists of multiple file systems that form virtual storage space for multiple users. The file system's organization is a hierarchical structure similar to an inverted tree; that is, a branching structure where top-level files contain other files, which in turn contain other files. Figure 2-1 illustrates a typical UNIX directory.

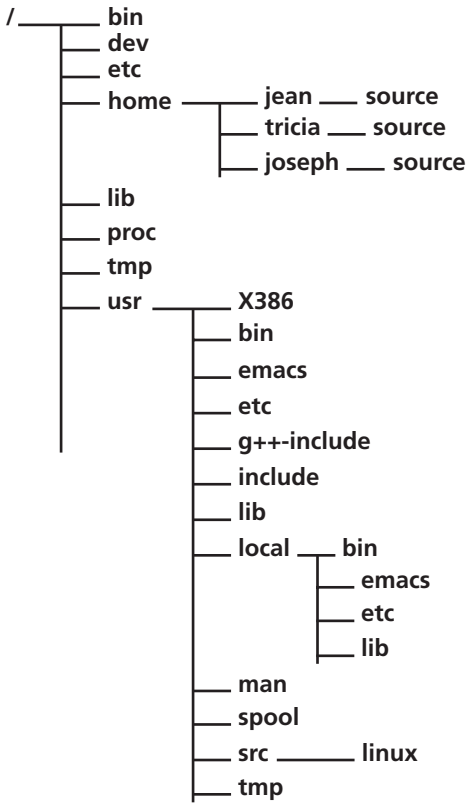


Figure 2-1: Typical UNIX directory

Understanding the Standard Tree Structure

The tree-like structure for UNIX file systems starts at the root level. **Root** is the name of the file at this basic level, and it is denoted by the slash character (/). The slash represents the **root directory**.

A **directory** is a special kind of file that can contain other files and directories. Regular files store information, such as records of employee names and addresses or payroll information, while directory files store the names of regular files and the names of other directories, which are called **subdirectories**. The subdirectory is considered the **child** of the **parent** directory because the child directory is created within the parent directory. In Figure 2-1, the root directory (/) is the parent of all the other directories. The home directory, for example, is the parent of the jean, tricia, and joseph subdirectories.

Using UNIX Partitions

The section of the disk that holds a file system is called a **partition**. One disk may have many partitions, each separated from the others so that it remains unaffected by external disturbances such as structural file problems associated with another partition. When you install UNIX on your computer, one of your first tasks is deciding how to partition your hard drive.

UNIX partitions are identified with names such as "hda1" and "hda2." The first two letters tell UNIX the device type: "hd," for instance, means hard disk. The third letter, "a," for instance, indicates whether the disk is the primary or secondary disk (a=primary, b=secondary). Partitions on a disk are numbered starting with 1. The name "hda1" tells UNIX that this is the first partition on the disk, and the name "hda2" indicates it is the second partition on the same disk. If you have a second hard disk with two partitions, the partitions are identified as "hdb1" and "hdb2." Computer storage devices such as hard disks are called peripheral devices. Computer **peripherals**, like hard disks, connect to the computer through **electronic interfaces**. The two most popular hard disk interfaces are the **IDE** (integrated drive electronics) and **SCSI** (small computer system interfaces). On PCs, IDE hard disk drives are more common than SCSI (pronounced "scuzzy"). The SCSI is faster and more commonly used on local-area-network servers. If you have a primary SCSI hard disk with two partitions, the two partitions are named "sda1" and "sda2." Figure 2-2 shows two partition tables: one with an IDE drive and the other with a SCSI drive.

```
Disk/dev/hda: 128 heads, 63 sectors, 767 cylinders
Units = cylinders of 8064 * 512 bytes

    Device  Boot  Begin     Start    End   Blocks   Id  System
/dev/hda1    *        1       1    242   975712+   6  DOS 32-bit >=32M
/dev/hda2           243     243    767  2116899    5  Extended
/dev/hda3           243     243    275   127024+  83  Linux native
/dev/hda6           276     276    750  1028224+  83  Linux native
/dev/hda7           751     751    767    68512+  82  Linux swap

Command (m for help):  _
```

This partition table is from a Linux system with an IDE drive

```
Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes

    Device  Boot  Begin    Start     End    Blocks    Id  System
/dev/sda1            1        1      64    514048+   83  Linux native
/dev/sda2           65       65    1106   8369865     5  Extended
/dev/sda5           65       65    1084   8193118+   83  Linux native
/dev/sda6         1085     1085    1100    128488+   82  Linux swap

Command (m for help):  _
```

This partition table is from a Linux system with a SCSI drive

Figure 2-2: Two partition tables

Note that the first table in Figure 2-2 identifies "hda" as the device, which indicates an IDE drive. The second table identifies "sda" as the device, which indicates a SCSI drive.

Setting Up File System Partitions

Partitioning your file systems protects and insulates each file system. If one file system fails, you can work with another. This section provides general guidelines on how to partition hard disks. These recommendations are suggestions only. How you partition your hard drive may vary depending on your system's configuration, number of users, and planned use. Partition size is measured in megabytes (MB, a million characters). You should have at least three separate partitions to store your file systems: root, swap, and /usr.

Begin by setting up a partition for the root file system. A partition must be mounted before it becomes part of the file system. The kernel mounts the root file system when the system starts. The kernel has no other information about the computer's other file systems, so all the information and programs the kernel needs to start the system must be present on the root partition. To protect this critical information, isolate the root partition from the other partitions.

Large partitions have a greater chance of being corrupted than small partitions do. The root partition should be small to avoid file corruption. A corrupted root file system makes the system unbootable. Red Hat and most Linux distributors recommend a root partition size of no more than 120 MB.

After creating the root partition, you should set up the **swap** partition. The swap partition acts like an extension of memory, so that UNIX has more room to run large programs. Linux distributors suggest that your swap space be twice the size of your computer's internal memory (RAM). For example, if your computer has 16 MB of RAM, you should allocate 32 MB for the swap partition. Linux distributors recommend not allocating more than 128 MB of swap space, because doing so wastes space. See Appendix C, "Installing Red Hat 5.x," for more information.

The **/usr** partition stores all the non-kernel operating system programs that make the computer useful. These programs include software development packages that support computer programming, networking, Internet access, graphical screens (including X-Windows), and the large number of UNIX **utilities** (programs that perform utilitarian operations like copying files, listing directories, and communicating with other users). The usr partition should be the largest of the three partitions, at least 600 MB.

In addition to these three separate partitions, you can create a **/home** partition, the home directory for all users' directories. A separate home partition protects and insulates users' personal files from the UNIX operating system software.

For example, Dominion Consulting partitions a PC with 16 MB of RAM and allocates 2 GB of hard disk space for UNIX, allowing 120 MB for the root partition, 1400 MB for the /usr partition, 32 MB for the swap partition, and 448 MB for the /home partition.

The /usr Partition

The /usr partition is a large partition that stores most operating system files and programs. Most Linux distributors, including Red Hat, offer you the option to "install everything" from their CD-ROM and store most of these files in the /usr partition. This is true of other UNIX systems as well. Software development tools such as language compilers, shared libraries, and header files needed for program creation are also stored in a hierarchy of subdirectories starting at /usr.

The /home Partition

The /home partition is the storage space for all users' work. If the /root partition (or any other partition) crashes, having a /home partition ensures that you do not lose all the users' information. Although you are restricted from reading information in other partitions, you own and can access most files in your home directory. You can grant or deny access to your files as you choose. See "Setting File Permissions" later in this chapter for more information on file ownership.

The size of the partition on a system for all users' home directories depends on how many users the system supports. Thirty users probably need a total of 3000 MB, that is, 3 GB (3 billion characters). This scheme gives users 100 MB each to store their work. The space allocated depends on how much data each user needs to store.

The /swap Partition

Swap partitions support **virtual memory**. Virtual memory means you have an unlimited memory resource—swap partitions try to meet this goal by providing swap space on disk and treating it like an extension of memory (RAM). It is called swap space because the system can use it to swap information between disk and RAM. Setting up swap space makes your computer run faster and more efficiently. If your computer has 16 MB of RAM or less, you must allocate some disk space to a swap partition. Even if your computer has more than 16 MB of RAM, swap space is recommended for all UNIX systems. The minimum amount of swap space should equal your RAM.

The largest possible swap space is 127 MB on Linux systems; any additional space is wasted. You can create and use more than one swap partition if you want more than 127 MB, although this is only necessary for large server installations.

Exploring the Root File System

The root (/) file system is mounted by the kernel when the system starts. To **mount** a file system is to connect it to the directory tree structure. The system administrator uses the mount command to mount a file system. The syntax of the mount command is:

Syntax	**mount** *device-name mount-point*
Dissection	■ ***device-name*** identifies the partition (file system) to mount. ■ ***mount-point*** identifies the directory where you want to mount the file system.

See "Using the mount Command" later in this chapter for more information about the mount command.

You must mount a file system before the system can access it. After mounting, the root file system is accessible for reading only. The root file system contains all essential programs for file system repair, restoring from a backup, starting the system, and initializing all devices and operating resources. It also contains the information for mounting all other file systems. Nothing beyond these essentials should reside in the root partition.

You can restore a "crashed" root partition using rescue files stored on floppy disks or tape. The installation media that comes with Red Hat Linux creates rescue disks.

The root directory itself generally contains subdirectories that contain files. The next sections describe the more frequently referenced subdirectories of the root file system.

The /bin Directory

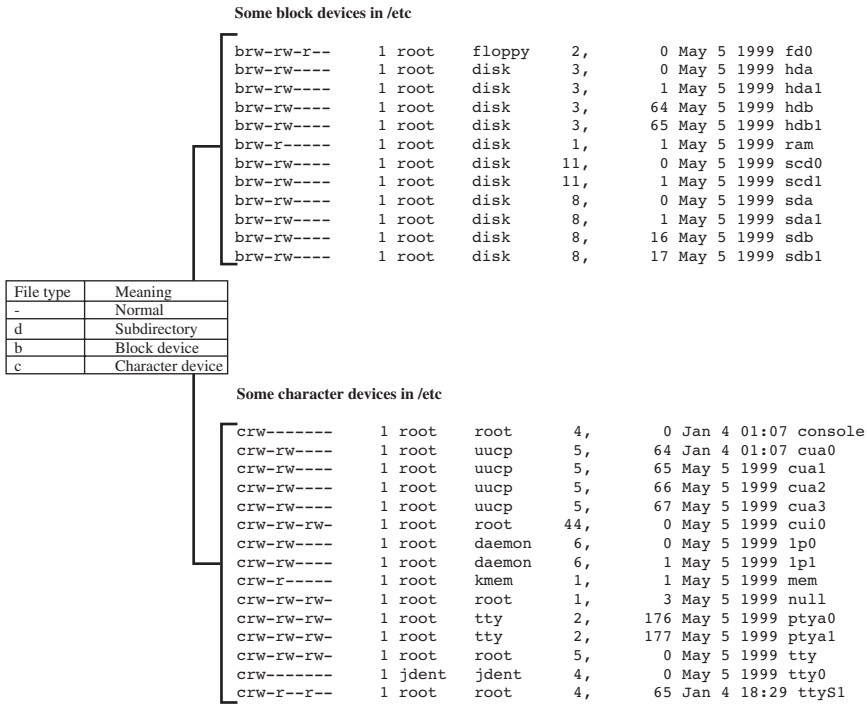
The /bin directory contains **binaries**, or **executables**, the programs needed to start the system and perform other essential system tasks. This directory holds many programs that all users need to work with UNIX.

The /boot Directory

The /boot directory often contains the files that the **bootstrap loader** (the utility that starts the operating system) needs and the kernel (operating system) images.

The /dev Directory

Files in /dev are device drivers. They access system devices and resources such as hard disks, the mouse, printers, consoles, modems, memory, floppy disks, and the CD-ROM drive. All UNIX versions include many device files in the /dev directory to accommodate separate vendor devices that can be attached to the computer. The device files are divided into two major classifications: block types and character types. The type indicates the method of data transmission on the device, either as a block of characters or as a serial flow of characters. You can see the list of device files by typing ls –l /dev after the command prompt. (See "Listing Directory Contents" later in this chapter for more information on the ls command.) The leftmost character on the list tells you whether the file is a character device (c) or a block device (b) as shown in Figure 2-3.



Some block devices in /etc

```
brw-rw-r--     1 root    floppy    2,       0 May 5 1999 fd0
brw-rw----     1 root    disk      3,       0 May 5 1999 hda
brw-rw----     1 root    disk      3,       1 May 5 1999 hda1
brw-rw----     1 root    disk      3,      64 May 5 1999 hdb
brw-rw----     1 root    disk      3,      65 May 5 1999 hdb1
brw-r-----     1 root    disk      1,       1 May 5 1999 ram
brw-rw----     1 root    disk     11,       0 May 5 1999 scd0
brw-rw----     1 root    disk     11,       1 May 5 1999 scd1
brw-rw----     1 root    disk      8,       0 May 5 1999 sda
brw-rw----     1 root    disk      8,       1 May 5 1999 sda1
brw-rw----     1 root    disk      8,      16 May 5 1999 sdb
brw-rw----     1 root    disk      8,      17 May 5 1999 sdb1
```

File type	Meaning
-	Normal
d	Subdirectory
b	Block device
c	Character device

Some character devices in /etc

```
crw-------     1 root    root      4,       0 Jan 4 01:07 console
crw-rw----     1 root    uucp      5,      64 Jan 4 01:07 cua0
crw-rw----     1 root    uucp      5,      65 May 5 1999 cua1
crw-rw----     1 root    uucp      5,      66 May 5 1999 cua2
crw-rw----     1 root    uucp      5,      67 May 5 1999 cua3
crw-rw-rw-     1 root    root     44,       0 May 5 1999 cui0
crw-rw----     1 root    daemon    6,       0 May 5 1999 lp0
crw-rw----     1 root    daemon    6,       1 May 5 1999 lp1
crw-r-----     1 root    kmem      1,       1 May 5 1999 mem
crw-rw-rw-     1 root    root      1,       3 May 5 1999 null
crw-rw-rw-     1 root    tty       2,     176 May 5 1999 ptya0
crw-rw-rw-     1 root    tty       2,     177 May 5 1999 ptya1
crw-rw-rw-     1 root    root      5,       0 May 5 1999 tty
crw-------     1 jdent   jdent     4,       0 May 5 1999 tty0
crw-r--r--     1 root    root      4,      65 Jan 4 18:29 ttyS1
```

Figure 2-3: Device files in /dev

Explanations of the items listed in Figure 2-3 follow:

- **console** refers to the system's console—that is, the monitor connected directly to your system.
- **ttyS1** and **cua1** are devices used to access serial ports. For example, /dev/ttyS1 refers to COM2, the communication port on your PC.
- **cua** devices are "callout" devices used in conjunction with a modem.
- Device names beginning with **hd** access hard drives.
- Device names beginning with **sd** are SCSI drives.
- Device names beginning with **lp** access parallel ports. The lp0 device refers to LPT1, the line printer.
- **null** is a "black hole"—any data sent to this device is gone forever. Use this device when you want to suppress the output of a command appearing on your screen. Chapters 6 and 7 discuss this technique.
- Device names beginning with **tty** refer to "virtual consoles" on your system. (You access them by pressing Alt+F1, Alt+F2, and so on.) Chapter 11 discusses virtual consoles, which let you switch between screens.
- Device names beginning with **pty** are "pseudo-terminals." They are used to provide a terminal to remote login sessions. For example, if your machine is on a network, incoming remote logins would use one of the pty devices in /dev.

The /etc Directory

The /etc directory contains configuration files the system uses when the computer starts. Most of this directory is reserved for the system administrator, and it contains system-critical information stored in files:

- **passwd**, the user database
- **rc**, scripts or directories of scripts to run when the system starts
- **fstab**, lists of file systems mounted automatically when the system starts
- **group**, the user group database
- **inittab**, the configuration file for the **init** program that performs essential chores when the system starts
- **motd**, the message of the day file
- **printcap**, the printer capability database
- **termcap**, the terminal capability database
- **profile** and **bashrc**, files executed at logon that let the system administrator set global defaults for all users
- **login.defs**, the configuration file for the **login** command

The /lib Directory

This directory houses the **shared library images**, files that programmers generally use to share code in the libraries rather than creating copies of this code in their programs. This makes the programs smaller and faster. Many files in this directory are symbolic links to files in system libraries. A **symbolic link** is a name that points to and lets you access a file located in a directory other than the current directory. In the directory's long listing, *l* in the leftmost position identifies files that are symbolic links.

The /mnt Directory

Mount points for temporary mounts by the system administrator reside in the /mnt directory. This directory is often divided into subdirectories such as /mnt/cdrom and /mnt/floppy, to clearly specify device types.

The /proc Directory

The /proc directory occupies no space on the disk: it is a **virtual file system** allocated in memory only. Files in /proc refer to various processes running on the system, so you can find information about which programs and processes are running.

The /root Directory

The /root directory is the home directory for the user root, usually the system administrator.

The /sbin Directory

The /sbin directory is reserved for the system administrator. Programs that start the system, programs needed for file system repair, and essential network programs are stored here.

The /tmp Directory

Many programs need a temporary place to store data during processing cycles. The traditional location for these files is the /tmp directory.

The /var Directory

The /var directory holds subdirectories whose sizes often change. These subdirectories contain files such as error logs and other system performance logs that are useful to the system administrator. The /var/spool/mail subdirectory contains mail from the network, which remains there until you read and delete it.

Using the mount Command

As you learned, UNIX uses the mount command to connect the file system partitions to the directory tree when the system starts. Users can access virtually anything on the system; only the total number of partitions mounted limits access. After the system starts, you can remove this limit by using the mount command to make other file systems accessible. The floppy disk and CD-ROM drives are the two devices that users most commonly need, so you must mount them.

To ensure security on the system, only the root user, usually the system administrator, can use the mount command. However, ordinary users can use several software packages to mount and unmount file systems, particularly on floppy disks and CDs.

Suppose you want to access files on a CD for the Dominion Consulting database. The system administrator mounts a CD-ROM by inserting a disk in the drive and uses the following mount command:

```
mount –t iso9660 –r /dev/cdrom /mnt/cdrom
```

This command mounts the CD on a device called “cdrom” located in the /dev directory. The actual mount point in UNIX is /mnt/cdrom, a directory that references the CD-ROM device. Once the CD is mounted, you can access its files through the /mnt/cdrom directory. UNIX supports several different types of file systems. The type of file system is specified with the –t option. CD-ROMs are classified as iso9660 devices, so the system administrator types –t, followed by the argument iso9660. The –r indicates that the CD-ROM device is read-only.

You also want to store back-up files on a floppy disk. The system administrator mounts a floppy disk in drive A using the command:

```
mount /dev/fd0 /mnt/floppy
```

This command mounts the floppy in drive A, which is the device /dev/fd0, to the mount point, /mnt/floppy, in the UNIX file structure. Any files stored in the directory /mnt/floppy are written to the floppy.

After accessing manually mounted file systems, the system administrator unmounts them using the umount command before removing the storage media, as in these examples:

```
umount /mnt/floppy
umount /mnt/cdrom
```

See Appendix B, "The UNIX Command Syntax Guide to the Linux Shell and Utility Programs," for a complete description of the mount and umount commands.

Understanding Paths and Pathnames

As you learned, all UNIX files are stored in directories in the file system, starting from the root directory. To specify a file or directory, use its **pathname**, which follows the branches of the file system to the desired file. A forward slash (/) separates each directory name. For example, suppose you want to specify the location of the file phones.502. You know that it resides in the source directory, in Jean's home directory, illustrated in Figure 2-1. You can specify this file's location as /home/jean/source/phones.502.

Using Your Command-line Prompt

The UNIX command prompt may indicate your location within the file system. For example, the prompt [jean@eli jean]$ is probably the **default prompt** that the system generated when the system administrator first created the login account. The prompt [jean@eli jean]$ means that "jean" is the user working on the host machine called "eli" in her home directory, which bears her user name, "jean." In other words, "jean is at eli in her home directory." When Jean changes her location to /home/jean/source, her prompt looks like:

```
[jean@eli source]
```

When the system is initially installed, the default root prompt looks like this: [root@spirit /root]#. To simplify the meaning of the command prompts in this book, the steps use $ to represent the ordinary user's command prompt and # to represent the system administrator's command prompt.

Customizing Your Prompt

Your login prompt is configured automatically when you login. An environment variable, PS1, contains special formatting characters that determine your prompt's configuration.

To see the contents of the PS1 environment variable:

1. Type **echo $PS1** and press **Enter**.
2. You see the contents of the PS1 variable, which may appear as:

```
[\u@\h \W]\$
```

Characters that begin with \ are special Bash shell formatting characters. \u prints the user name, \h prints the system host name, and \W prints the name of the working directory. The \$ character prints either a # or a $, depending on the type of user logged on. The brackets, [and], and the space that separates \h and \W, are not special characters, so they are printed just as they appear. When Jean is logged on to the system eli and working in her home directory, her prompt appears as [jean@eli jean]$ in the format shown above.

Table 2-1 shows other formatting characters for configuring your Bash shell prompt.

Formatting character	Purpose
\d	Displays the date
\h	Displays the host name
\n	Displays a new line
nnn	Displays the ASCII character that corresponds to the octal number *nnn*.
\s	Displays the shell name
\t	Displays the time
\u	Displays the user name
\w	Displays the path of the working directory
\W	Displays the name of the working directory without any other path information
\!	Displays the number of the current command in the command history

Table 2-1: Formatting characters for configuring a Bash shell prompt

Formatting character	Purpose
\#	Displays the number of the command in the current session
\$	Displays a # if root is the user, otherwise displays a $
\[	Marks the beginning of a sequence of non-printing characters, such as a control sequence
\]	Marks the end of a sequence of non-printing characters
\\	Displays a \ character

Table 2-1: Formatting characters for configuring a Bash shell prompt (continued)

To configure your Bash shell prompt:

1. To change your prompt to display the date and time, type **PS1='\d \t>'** and press **Enter**. Type the command with no spaces between the characters. Your prompt now looks similar to:

   Tue Jul 5 09:18:33>

2. To change your prompt to display the current working directory, type **PS1='\w>'** and press **Enter**. Your prompt now looks similar to:

   ~>

   The \w formatting character displays the ~ to represent the user's home directory.

   To change your prompt to display the full path of the current working directory, you must use another environment variable, PWD. The PWD variable contains the full pathname of the current working directory.

3. To display the PWD variable in the prompt, type **PS1='$PWD>'** and press **Enter**. (Notice that you must place the $ sign in front of the environment variable name to extract its contents.) Your prompt now looks similar to:

   /home/jean>

4. Log out of the system, and log back on to reset your prompt to its default configuration.

The pwd Command

You can use the UNIX pwd command to display your current path (pwd stands for **print working directory**).

To display your current path:

- After the $ command prompt, type **pwd** and press **Enter.** The system displays the path of your current working directory.

Navigating the File System

To navigate the UNIX directory structure, use the cd (change directory) command. Its syntax is:

Syntax	cd *directory*
Dissection	■ *directory* is the name of the directory to which you want to change. The directory name is expressed as a path to the destination, with slashes (/) separating subdirectory names.

When you log on, you begin in your home directory, which is under the /home directory. When you change directories and then want to return to your home directory, type cd and press Enter. (UNIX also uses the tilde character (~) to denote the user's home directory.)

You want to go to the mail subdirectory to see if Dominion employees sent you e-mail.

To change directories:

1. After the $ command prompt, type **cd /etc/mail** and press **Enter**. This moves you to the /etc/mail subdirectory.
2. Type **cd** and press **Enter**. The change directory command (cd) without arguments returns you to your home directory.

UNIX refers to a path as either an **absolute path** or a **relative path.** An **absolute path** begins at the root level and lists all subdirectories to the destination file. For example, assume that Becky has a directory named lists, located under her home directory. In the lists directory she has a file called todo. The absolute path to the todo file is \home\becky\lists\todo. The path lists each directory that lies in the path to the todo file.

Any time the \ symbol is the first character in a path, it stands for the root directory. All other \ symbols in a path serve to separate the other names.

A **relative path** takes a shorter journey. You can enter the relative path to begin at your current working directory and proceed from there. In Figure 2-1, jean, tricia,

and joseph each have subdirectories located in their /home directories. Each has a subdirectory called "source." Because Jean is working in her home directory, she can change to her source directory by typing the following command and pressing Enter:

```
cd source
```

Jean is changing to her source directory directly from her home directory, /home/jean. This example uses relative path addressing. Her source directory is one level away from her current location, /home/jean. As soon as she enters the change directory command, cd source, the system takes her to /home/jean/source, because it is relative to her current location.

If Tricia, who is in the /home/tricia directory, enters the command cd source, the system takes her to the /home/tricia/source directory. For Tricia to change to Jean's source directory, she can enter:

```
cd /home/jean/source
```

This example uses absolute path addressing, because Tricia starts from the root directory and works through all intervening directories.

To navigate directories:

1. If you are not in your home directory, type **cd** and press **Enter**.
2. The parent directory of your home directory is /home. /home is an absolute path name. Type **cd /home** and press **Enter**. The system takes you to the /home directory.

   You may now use the relative path with the cd command to return to your home directory. For example, if your user name is phillip, you can type cd phillip.
3. Type the **cd** command followed by your user name, and press **Enter**. The system takes you to your home directory.

Using Dot and Dot Dot Addressing Techniques

UNIX interprets a single **dot** character to mean the current directory, and **dot dot** (two consecutive dots) to mean the parent directory. Entering the following command keeps you in the current directory:

```
cd .
```

If you use two dots (known as dot dot), you move back to the parent directory. Do not type a space between the two dots. The next example shows how the user, jean, returns to her home directory, which is /home/jean:

```
cd ..
```

Assume you are Jean in her home directory and want to go to Tricia's source directory. Use the following command:

```
cd ../tricia/source
```

Although it is not always required, including a space after all UNIX commands is good practice if the commands include arguments.

In the previous example, the dot dot tells the operating system to go to the parent directory, which is /home. The first / separator followed by the directory name tells the operating system to go forward to the tricia subdirectory. The second "/" separator followed by the directory name tells UNIX to go forward to the source subdirectory, the final destination. If no name follows the slash character, UNIX treats it as the root directory. Otherwise, / separates one directory from another.

To use dot and dot dot to change your working directory:

1. If you are not in your home directory, type **cd** and press **Enter**.
2. Type **cd .** and press **Enter**. Since . references your current directory, the system did not change your working location.
3. Type **cd ..** and press **Enter**. The system takes you to the parent directory, which is /home.
4. Type **cd ..** and press **Enter**. The system takes you to the root (/) directory.
5. Type **cd** and press **Enter**. The system takes you to your home directory.

Listing Directory Contents

Use the ls (list) command to display a directory's contents, including files and other directories. When you use the ls command with no options or argument, it displays the names of all files and directories in your current working directory.

To see a list of files and directories in your current working directory:

- Type **ls** and press **Enter**.
  You see a list of file and/or directory names.

You can provide an argument to the ls command to see the listing for a specific file or to see the contents of a specific directory.

To see a listing for a specific file or directory:

1. If you are not in your home directory, type **cd** and press **Enter**.
2. In Chapter 1, you used the cat command to create a notes file. You should still have that file in your home directory. Type **ls notes** and press **Enter**. The system displays the listing for the notes file.

3 To see the contents of a directory other than your current working directory, give the directory name as an option to the ls command. For example, to see the contents of the /var directory, type **ls /var** and press **Enter**. You see a listing similar to the one below.

```
catman db gdm lib local lock log nis preserve run spool tmp yp
```

You can also use options to display specific information or more information than the command alone provides. The –l option for the ls command generates a long directory listing, which includes complete information about each file. You decide to print a long listing of the /etc and /home directories for the Dominion system.

To use the ls command with the –l option:

1 Type **ls -l /etc** and press **Enter**.

You see information similar to that in Figure 2-4.

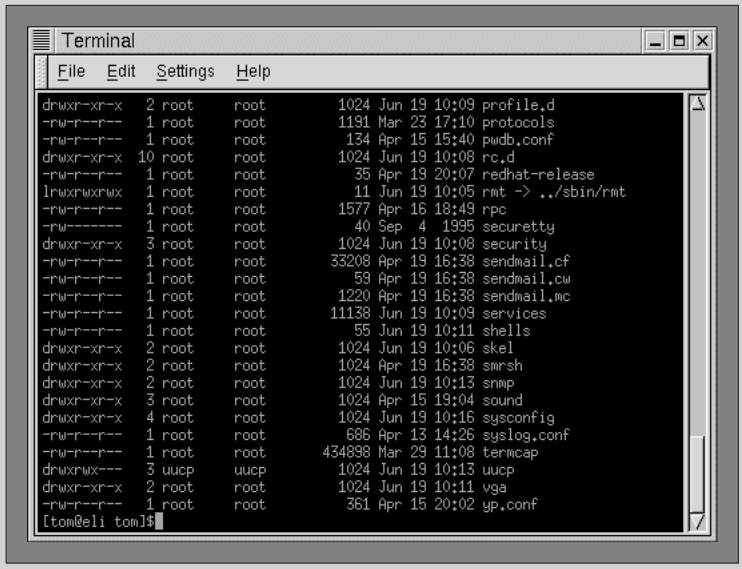


Figure 2-4: Contents of /etc directory

2 Type **ls –l /** and press **Enter**.

You see information similar to that in Figure 2-5.

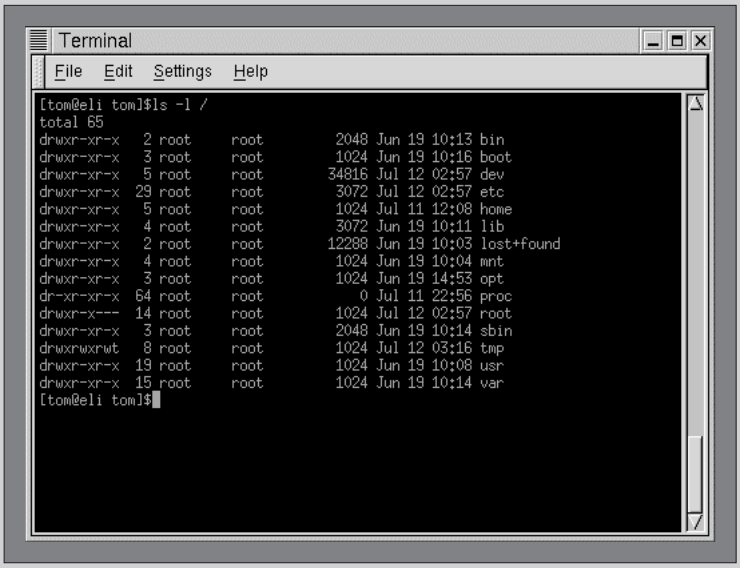


Figure 2-5: Contents of / directory

As you can see from the figures, the ls -l command provides more information about each item in the listing than a simple ls command. For example, look at the first item listed in Figure 2-5:

```
drwxr-xr-x    2 root    root        2048 Jun 19  10:13 bin
```

If you look in the rightmost column, you see bin, the name of a file. All the columns to its left contain information about the file bin. Here is a description of the information in each column, from left to right.

- *File Type and Access Permissions*: The first column of information shown is the following set of characters:

  drwxr-xr-x

  The first character in the list, d, indicates the file is actually a directory. If bin were an ordinary file, a hyphen (-) would appear instead. The remainder of the characters indicate the file's access permissions. You will learn more about these later in this chapter, in the section, "Setting File Permissions."
- *Number of Links*: The second column is the number of files that are symbolically linked to this file. (You will learn about symbolic links in Chapter 5.) If the file is a directory, this is the number of subdirectories it contains. The listing for bin shows it contains two subdirectories. (A directory always contains at least two subdirectories: dot and dot dot.)

- *Owner*: The third column is the owner of the file. The root user owns the bin directory.
- *Group*: The fourth column is the group that owns the file. The root group owns the bin directory.
- *Size*: The fifth column shows the size of the file in bytes.
- *Date and Time*: The sixth and seventh columns show the date and time the file was created or last modified.
- *Name*: The eighth column shows the file's name.

You can also use the –a option with the ls command to list **hidden files,** those whose names begin with a dot. The operating system normally uses hidden files to keep configuration information and for other purposes. The system administrator at Dominion tells you that your home directory contains a number of hidden files. You can list them using the –a option with the ls command.

To list hidden files in your home directory:

- Type **ls –a** after the command prompt, and press **Enter**.

  You see a list similar to the one in Figure 2-6.

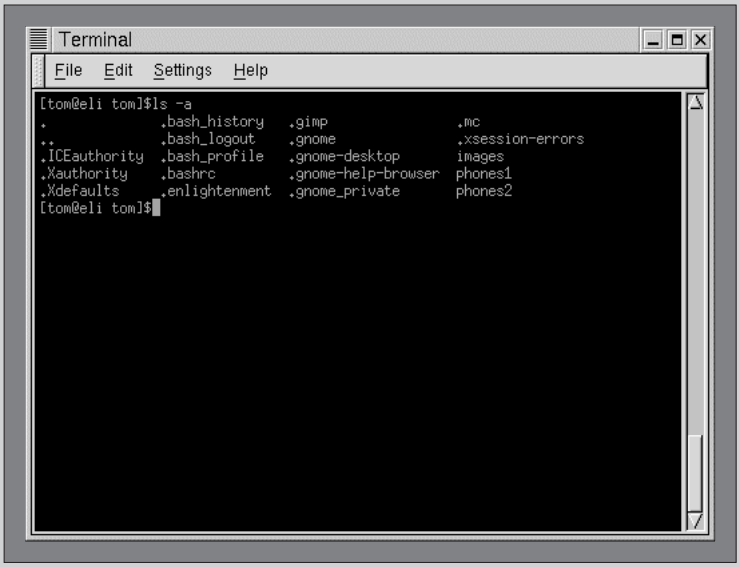


Figure 2-6: List of hidden files in the user's home directory

See Appendix B, "The UNIX Command Syntax Guide to the Linux Shell and Utility Programs," for a complete description of the ls command.

Using Wildcards

A wildcard is a special character that can stand for any other character or, in some cases, a group of characters. Wildcards are useful when you wish to work with several files whose names are similar or with a file whose exact name you cannot remember. UNIX supports several wildcard characters. In this section you will learn about two: * and ?.

The * wildcard represents any group of characters in a filename. For example, assume Becky has these nine files in her home directory:

```
friends
instructions.txt
list1
list2
list2b
memo_to_fred
memo_to_jill
minutes.txt
notes
```

If she enters ls *.txt and presses Enter, she sees the following output:

```
instructions.txt  minutes.txt
```

The argument *.txt causes ls to display the names of all files that end with .txt. If she enters ls memo*, she sees the following output:

```
memo_to_fred  memo_to_jill
```

If she enters the command ls *s and presses Enter, ls displays all filenames that end with s. She sees the output:

```
friends  notes
```

The ? wildcard only takes the place of a single character. For example, if Becky types ls list? and presses Enter, ls displays all files whose names start with *list* followed by a single character. She sees the output:

```
list1  list2
```

She does not see the listing for the file list2b, because two characters follow the word "list" in its name.

To work with wildcards:

1. To practice using wildcards, you first must create a set of files with similar names. In Chapter 1 you used the cat command to create the notes file. Use the cat command now to create these five files:

   first_name: a file containing your first name
   middle_name: a file containing your middle name
   last_name: a file containing your last name
   full_name1.txt: a file containing your full name
   full_name22 .txt: another file containing your full name

2. Type **ls *name** and press **Enter**. You see first_name, middle_name, and last_name listed.

3. Type **ls full_name?.txt** and press **Enter**. You see full_name1.txt listed.

LESSON B

objectives

After studying this lesson, you should be able to:

- Create new directories to store files
- Copy files from one directory to another
- Set file permissions to let other users access your directory and files

Working with Files and Directories

Creating Directories and Files

As part of your work to create Dominion's centralized telephone database, your manager at Dominion, Rolfe Williams, asks you to create directories for departments 4540 and 4550 and then create files of department phone numbers to store in those directories. You can use the mkdir (make directory) command to create a new directory and the cat command to create the phone files.

To create new directories and phone files:

1. Type **cd** and press **Enter** to make sure you are in your home directory.
2. Type **mkdir dept_4540** and press **Enter** to make a new directory called dept_4540.
3. Type **ls** and press **Enter**. You see the dept_4540 directory in the listing.
4. Type **cd dept_4540** and press **Enter** to change to the new directory. Now you can use the cat command to create a file called phones1. The phones1 file contains fields for area code, phone prefix, phone number, last name, and first name. A colon (:) separates each field.
5. Type these commands, pressing **Enter** at the end of each line:

   ```
   cat > phones1
   219:432:4567:Harrison:Joel
   219:432:4587:Mitchell:Barbara
   219:432:4589:Olson:Timothy
   ```

6. Press **Ctrl+Z**.
7. Type **cd** and press **Enter** to return to your /home directory.

8. Type **mkdir dept_4550** and press **Enter** to make a new directory called dept_4550.
9. Type **ls** and press **Enter**. You see the dept_4550 directory in the listing.
10. Type **cd dept_4550** and press **Enter** to change to the new directory. Now you can use the cat command to create the file phones2, which contains the same fields as the phones1 file.
11. Type these commands, pressing **Enter** at the end of each line:

```
cat > phones2
219:432:4591:Moore:Sarah
219:432:4567:Polk:John
219:432:4501:Robinson:Lisa
```

12. Press **Ctrl+Z**.

You created two new directories using the mkdir command and then created two files of phone number information using the cat command.

Note: You can delete empty directories by using the remove directory (rmdir) command. First, use the cd command to change to the parent directory of the subdirectory you want to delete. For example, if you want to delete the old directory in /home/old, first change to the home directory. Then type rmdir directory, as in rmdir old, and press Enter.

Copying Files

You use the UNIX copy command, cp, to copy files from one directory to another. The –i option warns you that the cp command will overwrite the destination file. You can also use the dot location (current directory) as shorthand to specify the destination of a cp command.

After you create the phone file, Rolfe wants you to create a new central directory for Dominion called corp_db and then copy the phones1 file into it. You can use the tilde character (~) to represent the location of your home directory.

To copy the phones1 file into a new directory, corp_db:

1. Type **cd** and then press **Enter** to return to the /home directory.
2. Type **mkdir corp_db** and press **Enter** to make a new directory.
3. Type **cd corp_db** and press **Enter** to change to the new directory.
4. To copy the phones1 file from the dept_4540 directory to the current directory, type **cp ~/dept_4540/phones1 .** and press **Enter**.
5. To copy the phones2 file from the dept_4550 directory to the current directory, type **cp ~/dept_4550/phones2 .** and press **Enter**.

Now the Dominion central database contains two files. You can use the cat command to combine the two files and redirect (>) the output to the corp_phones file, which lists all phone number information for the company.

To concatenate two files to one:

1. Type **cat phones1 phones2 > corp_phones** and press **Enter** to add the contents of the two phone files to one new file called corp_phones.
2. Type **more corp_phones** and press **Enter** to view the new file's contents.

   As you recall from Chapter 1, the more command, which lets you display files one screen at a time, is especially useful for reading long files.

Note: To delete files you do not need, use the remove (rm) command. First, use the cd command to change to the directory containing the file you want to delete. Then type rm *filename*. For example, to delete the file old in the current directory, type rm old. You receive a warning before the file is deleted, so be sure you want to remove a file permanently before using this command. You will learn more about the rm command in Chapter 4.

Setting File Permissions

Because UNIX is a multi-user system, users can set permissions for files they own so that others can read, write, or execute their files. A file's owner is the person who created it. The permissions the owner sets are listed as part of the file description. Figure 2-7 shows directory listings that describe file types.

File type	Meaning
-	Normal file
d	Subdirectory

```
Excerpt from ls -l /etc

drwxr-xr-x   12  root   root         1024   Feb   6    1996   X11
-rw-r--r--    1  root   root           10   Oct  15   19:11   adjtime
drwxr-xr-x    1  root   root         1024   Feb  27    2000   cron.daily
```

```
Excerpt from ls -l /home/jean/source

rw-rw-r--     1 jean    jean          387   Dec  12   23:11   phones.502
```

Figure 2-7: File types described in directory listings

Notice the long listing of the two directories. (Remember that the directory is just another file.) The earlier section, "Listing Directory Contents," describes the information presented in a long listing. Now you can look closer at the file permissions. For the first file described, the column on the far left shows the string of letters drwxr-xr-x. You already know the first character indicates the file type. The characters that follow are divided into three sections of file permission specifiers. There are three specifiers in each section, as illustrated in Figure 2-8.

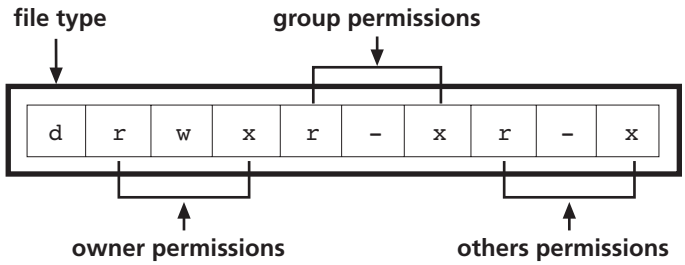


Figure 2-8: File permission specifiers

The first section of file permission specifiers indicates the owner's permissions. The owner, like all users, belongs to a group of users. The second section indicates the group's permissions. This specification applies to all users other than the owner who are members of the owner's group. The third section indicates all others' permissions. This specification applies to all users who are not the owner and not in the owner's group. In each section, the first character indicates read permissions. If an "r" appears there, that category of users has permission to read the file. The second character indicates write permission. If a "w" appears there, that category of user has permission to write to the file and delete the file. The third character indicates execute permission. If an "x" appears there, that category of user has permission to execute the file. If a dash (-) appears in any of these character positions, that type of permission is denied.

> **Note: If a user is granted read permission for a directory, the user can see a list of its contents. Write permission for a directory means the user can store and delete files in the directory. Execute permission for a directory means the user can make the directory the current working directory.**

From left to right, the letters drwxr-xr-x mean:

- d Indicates the file type (d = directory)
- r File's owner has read permission
- w File's owner has write permission
- x File's owner has execute permission (can run the file as a program)
- r Group has read permission
- - Group does not have write permission
- x Group has execute permission

r Others have read permission
- Others do not have write permission
x Others have execute permission

You can change the pattern of permission settings by replacing any of the three letters with a dash (-) to remove, or deny, permission. For example, setting others' permissions to - - - removes all permissions for others. They cannot read, write, or execute the file. In the first line of Figure 2-7, notice that the owner has **r**ead, **w**rite, and e**x**ecute (rwx) permissions for the file X11. The r-x indicates that the group of users that shares the same group id as the owner has only read and execute permissions. The system administrator assigns group ids when he or she adds a new user account. **Group ids** give a group of users equal access to files that they all share. Others are all other users who are not associated with the owner's group by a group id, but have read and execute permissions.

Use the UNIX chmod command to set file permissions. In its simplest form, the chmod command takes as arguments a symbolic string followed by one or more file names. The symbolic string specifies permissions that should be granted or denied to categories of users. Here is an example: ugo+rwx. In the string, the characters ugo stand for **u**ser (same as owner), **g**roup, and **o**thers. These categories of users will be affected by the chmod command. The next character, the + sign, indicates that permissions are being granted. The last set of characters, in this case rwx, indicates the permissions being granted. The symbolic string ugo+rwx indicates that read, write, and execute permissions are being granted to the owner, group, and others. Here is an example of how the symbolic string is used in a command, to modify the access permissions of myfile:

```
chmod ugo+rwx myfile
```

Here is a command that grants group read permission to the file customers:

```
chmod g+r customers
```

It is also possible to deny permissions with a symbolic string. The following command denies the group and others write and execute permissions for the file account_info.

```
chmod go-wx account_info
```

See Appendix B, "The UNIX Command Syntax Guide to the Linux Shell and Utility Programs," for a complete description of the chmod command.

Rolfe wants all users to have access to the corp_phones file. To make that possible, he asks you to change the file permissions. First, permit access to your home directory. Next, allow access to the corp_db directory, and then set the permissions for everyone to read the corp_phones file. You can use the execute permission command (x) to grant access to directories.

To change file and directory permissions:

1. Make sure that you are in the parent directory /home.
2. Type **chmod go+x ~** and press **Enter** to allow access to the home directory.

   This command means "make the home directory (~) accessible (+x) to the group (g) and others (o)."
3. Type **chmod ugo+x ~/corp_db** and press **Enter** to allow access to the corp_db directory.

   This command means "make the corp_db directory accessible (+x) for the owner (u), group (g), and others (o)."
4. Type **chmod ugo+r ~/corp_db/corp_phones** and press **Enter** to set permissions so everyone can read the file.

   This command means "make the corp_phones file readable (+r) for the owner (u), group (g), and others (o)."

Note: From your directory, you can create any subdirectory and set permissions for it. However, you cannot create subdirectories outside your home directory unless the system administrator makes a special provision.

You used the appropriate UNIX commands to create new directories to store files and set up a central directory for all users, and then you transferred files from other directories to the central directory. Finally, you changed file permissions so other users can access the directories and files you created.

SUMMARY

- In UNIX, a file is the basic component for data storage. UNIX considers everything to be a file, even attached devices such as the monitor, keyboard, and printer.
- A file system is the UNIX system's way of organizing files on mass storage devices such as hard and floppy disks. Files are stored on a file system, which is a hierarchical structure like a tree where top-level files contain other files, which in turn contain other files.
- Every file can be located by using a correct and unique pathname; that is, a listing of names of directories leading to a particular file.
- The standard tree structure starts with the root (/) directory, which serves as the foundation for a nested group of other directories and subdirectories.
- The section of the disk that holds the file system is called a partition. One disk may have many partitions, each separated from the others so that it remains unaffected by external disturbances such as structural file problems associated with another partition.
- The UNIX file system is a virtual file system, meaning that you can access all partitions once they are mounted in the tree structure.

- A path, as defined in UNIX, serves as a map to access any file on the system. An absolute path is one that always starts at the root level. A relative path is one that starts at your current location.
- You may customize your command prompt to display the current working directory name, the date, time, and several other items.
- The ls command displays the names of files and directories contained in a directory. The ls –l command and its options display all file information on the screen. This display is often called a long listing. The ls –a command shows hidden files.
- Wildcard characters can be used in a command, such as ls, and take the place of other characters in a file name. The * wildcard can take the place of any string of characters, and the ? wildcard can take the place of any single character.
- You can use the mkdir command to create a new directory as long as you own the parent directory. A file's owner is the person who creates it and becomes the only one who controls access to it.
- You can use the chmod command to set permissions for files that you own. The permissions settings are rwx, which mean read, write, and execute, respectively. File permissions are set to control file access by three types of users: the owner, the group, and others. You must remember to change permission settings on any directories you own if you want others to access information in those directories. You use the execute permission (x) command to grant access to directories.
- You can use the cp command to copy a source file to a destination file. UNIX overwrites the destination file without warning unless you use the –i option. The dot location (current directory) is a shorthand way to specify the destination in a cp command.

COMMAND SUMMARY

Chapter 2 commands

Command	Purpose	Options covered in this chapter
cd	Change directories	
chmod	Set file permissions for specified users	
cp	Copy files from one directory to another	-i prevents overwriting of the destination file without warning
ls	Display directory's contents, including its files and subdirectories	-a lists the hidden files -l generates a long listing of the directory

Chapter 2 commands		
mkdir	Make a new directory	
mount	Connect the file system partitions to the directory tree when the system starts	-r indicates that the mounted device is read-only -t specifies the type of file system
pwd	Display your current path	
rm	Remove a file	
rmdir	Remove an empty directory	
umount	Disconnect the file system partitions from the directory tree	

REVIEW QUESTIONS

1. A UNIX file system _____.
   a. is a directory
   b. is a physical allocation of space on a hard disk or other storage media
   c. has to be formatted to hold files
   d. a and c

2. The standard tree structure starts at _____.
   a. the current directory
   b. the root directory
   c. the first allocated partition
   d. the home directory

3. The absolute path must always start at _____.
   a. the current directory
   b. the root directory
   c. the first allocated partition
   d. the home directory

4. The most important reason to create a disk partition is to _____.
   a. make the system run faster
   b. conserve space on the disk
   c. protect information in that partition
   d. create adequate space to store application software

5. A virtual file system is one that _____.
   a. allows information to be retrieved from anywhere
   b. is created when all partitions are mounted
   c. lets users access virtually anything on the system; the total number of partitions mounted limits access
   d. all of the above
6. A partition named hda2 is _____.
   a. the first partition on the primary hard drive
   b. partition a on hard drive number 2
   c. the second partition on the primary hard drive
   d. the second partition on the secondary hard drive
7. A partition named sda1 is _____.
   a. the first partition on the secondary hard drive
   b. the first partition on the primary SCSI hard drive
   c. the first partition on the secondary SCSI hard drive
   d. none of the above
8. For safety purposes, the most critical file system that should be partitioned is _____.
   a. /
   b. /usr
   c. /home
   d. /bin
9. Most operating system files and programs are stored in the _____ partition.
   a. /
   b. /var
   c. /proc
   d. /usr
10. The _____ partition is the storage space for all users' work.
    a. /home
    b. /
    c. /root
    d. /bin
11. A path is _____.
    a. a tree structure
    b. a long directory listing
    c. a map of partitions
    d. a map to access any file on the system
12. Which of the following serves as virtual memory?
    a. the swap partition
    b. the / root partition
    c. the /etc partition
    d. the /usr partition

13. The /dev directory contains _____.
   a. device driver files
   b. development and programming utilities
   c. the UNIX bootstrap loader
   d. none of the above

14. The dot refers to _____.
   a. the parent directory
   b. the current directory
   c. the home directory
   d. the child directory

15. The dot dot refers _____.
   a. the parent directory
   b. the current directory
   c. the home directory
   d. the child directory

16. Which of these statements is false?
   a. You can create a child directory only in a parent directory you own.
   b. You must set the x permission for all directories that you want to access.
   c. You can use the cd .. command to move up one directory in the directory tree.
   d. The dot (.) refers to the parent directory.

17. You can use the dot as the location of the _____.
   a. parent directory
   b. child directory
   c. current directory
   d. root directory

18. The \h formatting character, used to customize the prompt, causes _____ to be displayed.
   a. the time in 24-hour format
   b. the host name
   c. the home directory name
   d. none of the above

19. The ls –l command is useful for checking _____.
   a. file permissions
   b. file type
   c. file size
   d. all of the above

20. The file type, shown in a long directory listing, for an ordinary file (not a directory) is the character _____.
   a. f
   b. d
   c. –
   d. space

21. Which of these grants the group read and write access to *filename*?
   a. chmod g+rw *filename*
   b. chmod g+r+w *filename*
   c. chmod o+rw *filename*
   d. chmod g-rw *filename*

22. Which command grants access to directories?
   a. chmod
   b. rwx
   c. execute permission (x)
   d. ls –x

23. The _____ environment variable stores the command-line prompt.
   a. PWD
   b. PROMPT
   c. PATH
   d. PS1

24. When you use the cp command, you should be aware that _____.
   a. the dot destination places the file in the parent directory
   b. the destination file will overlay a file with the same name
   c. it destroys the source file
   d. the source file cannot be relatively addressed

25. The remote directory is in the user's home directory. To copy a file from the remote directory to the current directory, type _____.
   a. cp ~/remote/*filename*
   b. cp +/*filename*
   c. cp > /remote/*filename*
   d. cp/remote/*filename*

26. The –i option, used with the cp command, causes _____.
   a. the cp command to ignore any requests not to overwrite the destination file, if it already exists
   b. the cp command to insert the file being copied into the existing destination file
   c. the cp command to warn you that it will overwrite the destination file
   d. none of the above

27. To list hidden files in a directory, type _____.
   a. hid -a
   b. ls –a
   c. cd\hidden\ls
   d. ls +a

EXERCISES

1. The file notice.txt is in the directory notices, which is in the directory public. The public directory is in Tom's home directory. (Tom's user name is tom.) What is the absolute path to the notice.txt file?
2. If your current working directory is the notices directory (under Tom's home directory), what is the relative path to the notices.txt file?
3. List an example of the cd command showing how to use absolute path addresses.
4. List an example of the cd command showing how to use relative path addresses. Log on and display the directories in /bin, /sbin, and /etc.
5. List the chmod symbolic string that will grant read and execute permission to a file's owner and group.
6. List the chmod symbolic string that will deny a file's read, write, and execute permission to all users who are not the owner or not in the owner's group.
7. Create a file using the cat command, and copy it to the /home directory. Change permissions so that no one can access that file.
8. Log on as another user, and try to access the files you previously created. Log on again under your account, and modify the files so they are accessible.

DISCOVERY EXERCISES

1. Use the ls command option to list hidden files in your home directory.
2. Use the absolute path to change to the /etc directory. From the /etc directory, change to the X11 directory using the relative path. Now list the .bashrc file in your home directory using the ls command. (Stay in the current location to do this.)
3. In the command line, type **cd ~/../..** and press **Enter**. Where are you now located in the tree structure? Explain how you got there.
4. In the command line, type **cd ~/../etc/X11** and press **Enter**. Where are you now located in the tree structure? Explain how you got there.
5. Using the relative path from your home directory, enter the cat command to display the contents of the /etc/passwd file.
6. Use the cd command to return to your home directory. Redirect the output of the date command to a file named chap2info. Use the ls command to get a long listing for this file. What permissions do the owner, group, and others have?
7. Change the permissions for the chap2info file to read and write for the owner and deny permissions for anyone else. Look at a long listing to confirm that the changes took effect.
8. Modify your command prompt so it always displays the absolute path of your current working directory.